Amendments to the Claims

Please amend claims 1, 6, 7, 15, 20, 21, 29, 35, 36, 47, 54, 55, 59, 60, 65, 74 and 75. A complete listing and status of the claims is as follows:

1. (Currently amended) A method of test generation for testing computer software, comprising the steps of:

responsively to a specification of a software application being verified, modeling ~~a~~ said software application as a finite state machine to define a behavioral model of said software application;

associating said behavioral model with a focus, said focus having a reference to said behavioral model, and having at least one directive; and

responsively to said directive, generating a test program for verification of said software application according to state transitions of said behavioral model ~~and said directive of said focus~~.

2. (Original) The method according to claim 1, wherein said directive comprises a model independent directive.

3. (Original) The method according to claim 1, wherein said directive comprises a model dependent directive, and a coverage variable of said behavioral model is tagged by a tag of said model dependent directive, said coverage variable having allowable values.

4. (Original) The method according to claim 3, wherein said directive further comprises a model independent directive.

5. (Original) The method according to claim 3, wherein said test program references said coverage variable, and said step of generating is performed until said coverage variable has assumed each of said allowable values.

6. (Currently amended) The method according to claim 5, wherein said coverage variable comprises a plurality of coverage variables, ~~and said step of generating is performed until a cross product of said coverage variables has assumed all possible values thereof~~ further comprising the step of defining a cross product of said coverage variables, said cross product having members formed by said allowable values of said coverage variables, wherein said step of generating is iterated to produce a plurality of test programs until each of said members is included in at least one of said test programs.

7. (Currently amended) The method according to claim 5, wherein said coverage variable comprises a plurality of coverage variables, ~~and said step of generating is performed until an orthogonal array of said coverage variables has assumed all possible values thereof~~ further comprising the step of defining an orthogonal array of said coverage variables, said orthogonal array having members formed by said allowable values of said coverage variables, wherein said step of generating is iterated to yield a plurality of test programs until each of said members is included in at least one of said test programs.

8. (Original) The method according to claim 3, wherein said model dependent directive comprises a plurality of model dependent directives, and said coverage variable is tagged by a plurality of tags of said model dependent directives.

9. (Original) The method according to claim 3, wherein said tag is a number-of-tests-per-value tag.

10. (Original) The method according to claim 3, wherein said model dependent directive is a mask-value directive.

11. (Original) The method according to claim 1, wherein said directive comprises a plurality of directives that are combined to define a directive expression, wherein said step of generating is performed until said directive expression has a predetermined value.

12. (Original) The method according to claim 1, wherein said step of modeling is performed by retrieving said behavioral model from a model archive.

13. (Original) The method according to claim 1, wherein said step of associating is performed by retrieving said focus from a focus archive.

14. (Original) The method according to claim 13, further comprising the steps of comparing state variables of foci of said focus archive with state variables of said behavioral model; and

responsive to comparisons resulting from said step of comparing revising selected ones of said foci.

15. (Currently amended) A computer software product, comprising a computer-readable medium in which computer program instructions are stored, which instructions, when read by a computer, cause the computer to execute a method of test generation for testing computer software, the method comprising the steps of:

accepting as a first input a behavioral model of a software application being verified, wherein said behavioral model is created responsively to a specification of said software application, and said behavioral model comprises a finite state machine;

accepting as a second input a focus having a reference to said behavioral model, and having at least one directive;

associating said behavioral model with said focus; and

responsively to said directive, generating a test program for verification of said software application according to state transitions of said behavioral model ~~and said directive of said focus~~.

16. (Original) The computer software product according to claim 15, wherein said directive comprises a model independent directive.

17. (Original) The computer software product according to claim 15, wherein said directive comprises a model dependent directive, and a coverage variable of said behavioral model is tagged by a tag of said model dependent directive, said coverage variable having allowable values.

18. (Original) The computer software product according to claim 17, wherein said directive further comprises a model independent directive.

19. (Original) The computer software product according to claim 17, wherein said test program references said coverage variable, and said step of generating is performed until said coverage variable has assumed each of said allowable values.

20. (Currently amended) The computer software product according to claim 19, ~~and said step of generating is performed until a cross product of said coverage variables has assumed all possible values thereof~~ further comprising the step of defining a cross product of said coverage variables, said cross product having members formed by said allowable values of said coverage

variables, wherein said step of generating is iterated to produce a plurality of test programs until each of said members is included in at least one of said test programs.

21. (Currently amended) The computer software product according to claim 19, ~~and said step of generating is performed until an orthogonal array of said coverage variables has assumed all possible values thereof~~ further comprising the step of defining an orthogonal array of said coverage variables, said orthogonal array having members formed by said allowable values of said coverage variables, wherein said step of generating is iterated to yield a plurality of test programs until each of said members is included in at least one of said test programs.

22. (Original) The computer software product according to claim 17, wherein said model dependent directive comprises a plurality of model dependent directives, and said coverage variable is tagged by a plurality of tags of said model dependent directives.

23. (Original) The computer software product according to claim 17, wherein said tag is a number-of-tests-per-value tag.

24. (Original) The computer software product according to claim 17, wherein said model dependent directive is a mask-value directive.

25. (Original) The computer software product according to claim 15, wherein said directive comprises a plurality of directives that are combined to define a directive expression, wherein said step of generating is performed until said directive expression has a predetermined value.

26. (Original) The computer software product according to claim 15, wherein said step of modeling is performed by retrieving said behavioral model from a model archive.

27. (Original) The computer software product according to claim 15, wherein said step of associating is performed by retrieving said focus from a focus archive.

28. (Original) The computer software product according to claim 27, further comprising the steps of comparing state variables of foci of said focus archive with state variables of said behavioral model; and

responsive to comparisons resulting from said step of comparing revising selected ones of said foci.

29. (Currently amended) A method of test generation for testing computer software, comprising the steps of:

responsively to a specification of a software application being verified, modeling ~~a~~ said software application as a finite state machine to define a behavioral model of said software application;

associating said behavioral model with a focus, said focus having a reference to said behavioral model, and having at least one directive;

responsively to said directive, deriving an abstract test suite for verification of said software application from said behavioral model and said focus, wherein said abstract test suite complies with a test constraint that is encoded in said focus;

executing said abstract test suite in an execution engine.

30. (Original) The method according to claim 29, wherein said step of executing said abstract test suite comprises the step of generating a test script from said abstract test suite; wherein said test script is executed in said execution engine.

31. (Original) The method according to claim 29, wherein said step of producing said abstract test suite is performed with a testing interface.

32. (Original) The method according to claim 31, wherein said testing interface comprises an abstract-to-concrete translation table.

33. (Original) The method according to claim 29, wherein said testing interface comprises a test driver, having an operator interface, and further comprising the step of:

varying parameters of said test driver via said operator interface in accordance with requirements of said software application.

34. (Original) The method according to claim 29, wherein said directive comprises a model independent directive.

35. (Currently amended) The method according to claim 38 wherein said coverage variable comprises a plurality of coverage variables, ~~and said step of generating is performed until a cross product of said coverage variables has assumed all possible values thereof~~ further comprising the step of defining a cross product of said coverage variables, said cross product having members formed by said allowable values of said coverage variables, wherein said step of generating is iterated to produce a plurality of test programs until each of said members is included in at least one of said test programs.

36. (Currently amended) The method according to claim 38, wherein said coverage variable comprises a plurality of coverage variables, ~~and said step of generating is performed until an orthogonal array of said coverage variables has assumed all possible values thereof~~ further comprising the step of defining an orthogonal array of said coverage variables, said orthogonal array having members formed by said allowable values of said coverage variables, wherein said step of generating is iterated to yield a plurality of test programs until each of said members is included in at least one of said test programs.

37. (Original) The method according to claim 29, wherein said directive comprises a model dependent directive, and a coverage variable of said behavioral model is tagged by a tag of said model dependent directive, said coverage variable having allowable values.

38. (Original) The method according to claim 37, wherein said abstract test suite references said coverage variable, and said step of generating is performed until said coverage variable has assumed each of said allowable values.

39. (Original) The method according to claim 37, wherein said directive further comprises a model independent directive.

40. (Original) The method according to claim 37, wherein said model dependent directive comprises a plurality of model dependent directives, and said coverage variable is tagged by a plurality of tags of said model dependent directives.

41. (Original) The method according to claim 37, wherein said tag is a number-of-tests-per-value tag.

42. (Original) The method according to claim 37, wherein said model dependent directive is a mask-value directive.

43. (Original) The method according to claim 29, wherein said directive comprises a plurality of directives that are combined to define a directive expression, wherein said step of generating is performed until said directive expression has a predetermined value.

44. (Original) The method according to claim 29, wherein said step of modeling is performed by retrieving said behavioral model from a model archive.

45. (Original) The method according to claim 29, wherein said step of associating is performed by retrieving said focus from a focus archive.

46. (Original) The method according to claim 29, further comprising the steps of comparing state variables of foci of said focus archive with state variables of said behavioral model; and

responsive to comparisons resulting from said step of comparing revising selected ones of said foci.

47. (Currently amended) A computer software product for testing computer software, comprising a computer-readable medium in which computer program instructions are stored, which instructions, when read by a computer, cause the computer to perform the steps of:

associating a behavioral model of a software application being verified with a focus, said focus having a reference to said behavioral model, and having at least one directive, wherein said behavioral model is created responsively to a specification of said software application, and said behavioral model ~~models~~ comprises a finite state machine;

responsively to said directive deriving an abstract test suite for verification of said software application from said behavioral model and said focus, wherein said abstract test suite complies with a test constraint that is encoded in said focus;

executing said abstract test suite in an execution engine.

48. (Original) The computer software product according to claim 47, wherein said step of executing said abstract test suite comprises the step of generating a test script from said abstract test suite; wherein said test script is executed in said execution engine.

49. (Original) The computer software product according to claim 47, wherein said step of producing said abstract test suite is performed with a testing interface.

50. (Original) The computer software product according to claim 49, wherein said testing interface comprises an abstract-to-concrete translation table.

51. (Original) The computer software product according to claim 49, wherein said testing interface comprises a test driver, having an operator interface, and further comprising the step of:

varying parameters of said test driver via said operator interface in accordance with requirements of said software application.

52. (Original) The computer software product according to claim 47, wherein said directive comprises a model independent directive.

53. (Original) The computer software product according to claim 29, wherein said directive comprises a model dependent directive, and a coverage variable of said behavioral model is tagged by a tag of said model dependent directive, said coverage variable having allowable values.

54. (Currently amended) The computer software product according to claim 53 wherein said coverage variable comprises a plurality of coverage variables, ~~and said step of generating is performed until a cross product of said coverage variables has assumed all possible values thereof~~ further comprising the step of defining a cross product of said coverage variables, said cross product having members formed by said allowable values of said coverage variables, wherein said step of generating is iterated to produce a plurality of test programs until each of said members is included in at least one of said test programs.

55. (Currently amended) The computer software product according to claim 53, wherein said coverage variable comprises a plurality of coverage variables, ~~and said step of generating is performed until an orthogonal array of said coverage variables has assumed all possible values thereof~~ further comprising the step of defining an orthogonal array of said coverage variables, said orthogonal array having members formed by said allowable values of said coverage variables, wherein said step of generating is iterated to yield a plurality of test programs until each of said members is included in at least one of said test programs.

56. (Original) The computer software product according to claim 53, wherein said abstract test suite references said coverage variable, and said step of generating is performed until said coverage variable has assumed each of said allowable values.

57. (Original) The computer software product according to claim 53, wherein said directive further comprises a model independent directive.

58. (Original) The computer software product according to claim 53, wherein said model dependent directive comprises a plurality of model dependent directives, and said coverage variable is tagged by a plurality of tags of said model dependent directives.

59. (Currently amended) The computer software product according to ~~claim 37, wherein said tag is~~ claim 58, wherein said tags comprise a number-of-tests-per-value tag.

60. (Currently amended) The computer software product according to ~~claim 37, wherein said model dependent directive is~~ claim 58, wherein said model dependent directive comprise a mask-value directive.

61. (Original) The computer software product according to claim 47, wherein said directive comprises a plurality of directives that are combined to define a directive expression, wherein said step of generating is performed until said directive expression has a predetermined value.

62. (Original) The computer software product according to claim 47, wherein said step of modeling is performed by retrieving said behavioral model from a model archive.

63. (Original) The computer software product according to claim 47, wherein said step of associating is performed by retrieving said focus from a focus archive.

64. (Original) The computer software product according to claim 63, further comprising the steps of comparing state variables of foci of said focus archive with state variables of said behavioral model; and

responsive to comparisons resulting from said step of comparing revising selected ones of said foci.

65. (Currently amended) A computer system for testing computer software, comprising:

a user interface for creating a behavioral model of a software application begin verified, responsively to a specification of said software application, said behavioral model representing a finite state machine, wherein said user interface creates a focus, said focus having a reference to said behavioral model, and having at least one directive;

a compiler, for converting said behavioral model into an intermediate encoding thereof;

a test generator, accepting said intermediate encoding and said focus as input, and responsively to said directive producing an abstract test suite for verifying said software application;

an execution engine for executing a test program of said abstract test suite.

66. (Original) The system according to claim 65, wherein said execution engine produces a suite execution trace.

67. (Original) The system according to claim 66, further comprising an analyzer which reads said suite execution trace, wherein said execution engine accepts an output of said analyzer.

68. (Original) The system according to claim 65, further comprising a visualizer for visualizing an output of said execution engine.

69. (Original) The system according to claim 65, wherein said execution engine further receives input from an application model interface that is created by said user interface.

70. (Original) The system according to claim 65, wherein said directive comprises a model independent directive.

71. (Original) The system according to claim 65, wherein said directive comprises a model dependent directive, and a coverage variable of said behavioral model is tagged by a tag of said model dependent directive, said coverage variable having allowable values.

72. (Original) The system according to claim 71, wherein said directive further comprises a model independent directive.

73. (Original) The system according to claim 71, wherein said test program references said coverage variable, and said test generator operates until said coverage variable has assumed each of said allowable values.

74. (Currently amended) The system according to claim 73, wherein said coverage variable comprises a plurality of coverage variables, ~~and said execution engine executes until a cross product of said coverage variables has assumed all possible values thereof~~ and said test generator is operative on an cross product of said coverage variables, said cross product having members formed by said allowable values of said coverage variables, wherein said test generator produces a plurality of test programs and each of said members is included in at least one of said test programs.

75. (Currently amended) The system according to claim 73, wherein said coverage variable comprises a plurality of coverage variables, ~~and said execution engine executes until an orthogonal array of said coverage variables has assumed all possible values thereof~~ and said test generator is operative on an orthogonal array of said coverage variables, said orthogonal array having members formed by said allowable values of said coverage variables, wherein said test generator produces a plurality of test programs and each of said members is included in at least one of said test programs.

76. (Original) The system according to claim 71, wherein said model dependent directive comprises a plurality of model dependent directives, and said coverage variable is tagged by a plurality of tags of said model dependent directives.

77. (Original) The system according to claim 71, wherein said tag is a number-of-tests-per-value tag.

78. (Original) The system according to claim 71, wherein said model dependent directive is a mask-value directive.

79. (Original) The system according to claim 65, wherein said directive comprises a plurality of directives that are combined to define a directive expression, wherein said execution engine executes until said directive expression has a predetermined value.

80. (Original) The system according to claim 65, further comprising a model archive that is accessed by said user interface.

81. (Original) The system according to claim 65, further comprising a focus archive that is accessed by said user interface.